Configuring Virtual Private LAN Service (VPLS) and VPLS BGP-Based Autodiscovery

- Restrictions for VPLS, on page 1
- Information About VPLS, VPLS BGP-Based Autodiscovery and Flow-Aware Transport, on page 1
- How to Configure VPLS, VPLS BGP-Based Autodiscovery and Flow-Aware Transport, on page 5
- Configuration Examples for VPLS and VPLS BGP-Based Autodiscovery, on page 25
- Feature History for VPLS and VPLS BGP-Based Autodiscovery, on page 30

Restrictions for VPLS

- Layer 2 protocol tunneling configuration is not supported
- Virtual Circuit Connectivity Verification (VCCV) ping with explicit null is not supported.
- The switch is supported if configured only as a spoke in hierarchical Virtual Private LAN Services (VPLS) and not as a hub.
- Integrated Routing and Bridging (IRB) configuration is not supported.
- Layer 2 VPN interworking functions are not supported.
- **ip unnumbered** command is not supported in Multiprotocol Label Switching (MPLS) configuration.
- Virtual Circuit (VC) statistics are not displayed for flood traffic in the output of **show mpls l2 vc vcid detail** command.
- Dot1q tunnel configuration is not supported in the attachment circuit.

Information About VPLS, VPLS BGP-Based Autodiscovery and Flow-Aware Transport

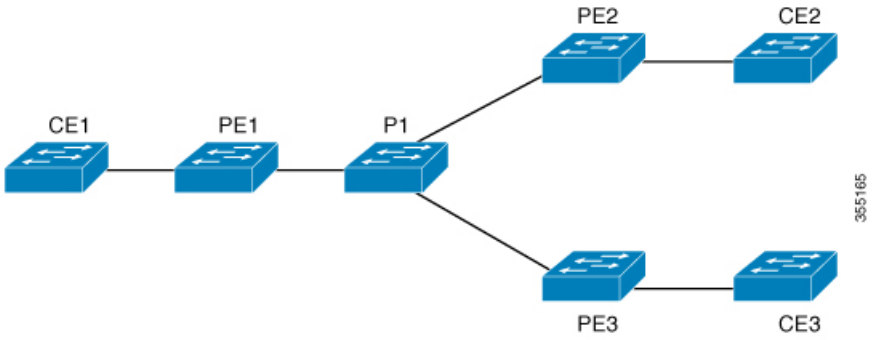
The following sections provide information about VPLS, VPLS BGP-based autodiscovery and flow-aware transport.

VPLS Overview

VPLS enables enterprises to link together their Ethernet-based LANs from multiple sites through the infrastructure provided by their service provider. From the enterprise perspective, the service provider's public network looks like one large Ethernet LAN. For the service provider, VPLS provides an opportunity to deploy another revenue-generating service on top of their existing network without major capital expenditures. Operators can extend the operational life of equipment in their network.

VPLS uses the provider core to join multiple attachment circuits together to simulate a virtual bridge between multiple attachment circuits. From a customer point of view, there is no topology for VPLS. All of the customer edge (CE) devices appear to connect to a logical bridge emulated by the provider core.

Figure 1: VPLS Topology



About Full-Mesh Configuration

The full-mesh configuration requires a full mesh of tunnel label switched paths (LSPs) between all the provider edge (PE) devices that participate in the VPLS. With full-mesh configuration, signaling overhead and packet replication requirements for each provisioned VC on a PE device are high.

For a full-mesh configuration, a virtual forwarding instance (VFI) is required on each participating PE device. The VFI includes the VPN ID of a VPLS domain, the addresses of other PE devices in the domain, and the type of tunnel signaling and encapsulation mechanism for each peer PE device.

A VPLS instance constitutes a set of VFIs formed by the interconnection of the emulated VCs. The VPLS instance forms the logic bridge over the packet switched network. The VPLS instance is assigned a unique VPN ID.

The PE devices use the VFI to establish a full-mesh LSP of emulated VCs to all the other PE devices in the VPLS instance. PE devices obtain the membership of a VPLS instance through the static configuration using the Cisco IOS CLI.

The full-mesh configuration allows the PE device to maintain a single broadcast domain. So when the PE device receives a broadcast, multicast, or unknown unicast packet on an attachment circuit, it sends the packet out on all other attachment circuits and emulated circuits, to all the other CE devices participating in that VPLS instance. The CE devices see the VPLS instance as an emulated LAN.

To avoid the problem of a packet looping in the provider core, the PE devices enforce a 'split-horizon' principle for the emulated VCs. The split-horizon principle ensures that a packet received on an emulated VC is not forwarded on any other emulated VC.

After the VFI has been defined, it needs to be bound to an attachment circuit to the CE device.

The packet forwarding decision is made by looking up the Layer 2 VFI of a particular VPLS domain.

A VPLS instance on a particular PE device receives Ethernet frames that enter on specific physical or logical ports and populates a MAC address table similarly to how an Ethernet switch works. The PE device uses the MAC address to switch those frames into the appropriate LSP, for delivery to the other PE device at a remote site.

If a MAC address is not populated in the MAC address table, the PE device replicates the Ethernet frame and floods it to all logical ports associated with that VPLS instance, except on the ingress port where the Ethernet frame had entered. The PE device updates the MAC address table as it receives packets on specific ports and removes addresses not used after specific periods.

About VPLS BGP-Based Autodiscovery

VPLS autodiscovery enables each PE device to discover other PE devices that are part of the same VPLS domain. VPLS autodiscovery also tracks PE devices when they are added to or removed from a VPLS domain. With VPLS autodiscovery enabled, it is no longer needed to manually configure a VPLS domain and maintain the configuration when a PE device is added or deleted. VPLS autodiscovery uses the Border Gateway Protocol (BGP) to discover VPLS members and set up and tear down pseudowires (PWs) in a VPLS domain.

BGP uses the Layer 2 VPN Routing Information Base (RIB) to store endpoint provisioning information, which is updated each time any Layer 2 VFI is configured. The prefix and path information is stored in the Layer 2 VPN database, which allows BGP to make decisions about the best path. When BGP distributes the endpoint provisioning information in an update message to all its BGP neighbors, this endpoint information is used to configure a pseudowire mesh to support Layer 2 VPN-based services.

The BGP autodiscovery mechanism facilitates the configuration of Layer 2 VPN services, which are an integral part of the VPLS feature. VPLS enables flexibility in deploying services by connecting geographically dispersed sites as a large LAN over high-speed Ethernet in a robust and scalable IP MPLS network.

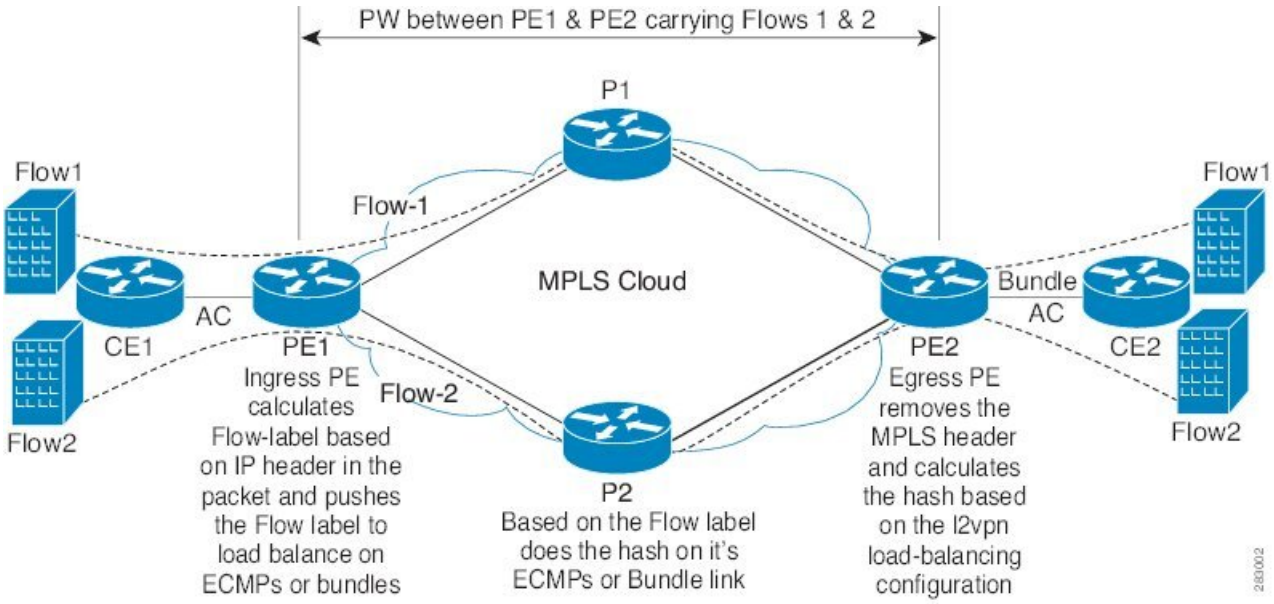
About Flow-Aware Transport Pseudowire

Devices typically load-balance traffic based on the lower most label in the label stack which is the same label for all flows on a given pseudowire. This can lead to asymmetric loadbalancing. The flow, in this context, refers to a sequence of packets that have the same source and destination pair. The packets are transported from a source provider edge (PE) device to a destination PE device.

Flow-aware transport PWs provide the capability to identify individual flows within a PW and provide devices the ability to use these flows to load-balance traffic. Flow-aware transport PWs are used to load-balance traffic in the core when equal cost multipaths (ECMP) are used. A flow label is created based on individual packet flows entering a PW; and is inserted as the lower most label in the packet. Devices can use the flow label for load-balancing which provides a better traffic distribution across ECMP paths or link-bundled paths in the core.

Figure 2: Flow-aware transport PW with two flows distributing over ECMPs and Bundle-Links shows a flow-aware transport PW with two flows distributing over ECMPs and bundle links.

Figure 2: Flow-aware transport PW with two flows distributing over ECMPs and Bundle-Links



An extra label is added to the stack, called the flow label, which contains the flow information of a virtual circuit (VC). A flow label is a unique identifier that distinguishes a flow within the PW, and is derived from source and destination MAC addresses, and source and destination IP addresses. The flow label contains the end of label stack (EOS) bit set and inserted after the VC label and before the control word (if any). The ingress PE calculates and forwards the flow label. The flow-aware transport PW configuration enables the flow label. The egress PE discards the flow label such that no decisions are made.

All core devices perform load balancing based on the flow-label in the flow-aware transport PW. Therefore, it is possible to distribute flows over ECMPs and link bundles.

Flow-aware transport PW works based on port-channel load-balance algorithm only.

Interoperability Between Cisco Catalyst 6000 Series Switches and Cisco Catalyst 9000 Series Switches

The following section describes how to enable sending and receiving flow labels between Cisco Catalyst 6000 Series Switches and Cisco Catalyst 9000 Series Switches.

On a Cisco Catalyst 6000 Series Switch configured with flow-aware transport PW (using Advanced VPLS) flow label negotiations are not supported. If the Cisco Catalyst 6000 Series Switch is in interoperability with a remote PE device such as a Cisco Catalyst 9000 Series Switch, then the Cisco Catalyst 9000 Series Switch cannot receive and send the flow label for data traffic. Configuring the **load-balance flow-label both static** command on the Cisco Catalyst 9000 Series Switch allows the Cisco Catalyst 9000 Series Switch to receive and send the flow labels even though the Cisco Catalyst 6000 Series Switch does not support flow label negotiations.

The following is a configuration example to enable sending and receiving flow labels:

```
Device> enable
Device# configure terminal
Device(config)# template type pseudowire mpls
Device(config-template)# encapsulation mpls
Device(config-template# load-balance flow ip dst-ip
```

```
Device(config-template)# load-balance flow-label both static
Device(config-template# end
```

IGMP or MLD Snooping over VPLS

Support for:

- IGMP Snooping over VPLS was introduced from Cisco IOS XE Amsterdam 17.1.1 release.
- MLD Snooping over VPLS was introduced from Cisco IOS XE Bengaluru 17.6.1 release.

By default:

- IGMP snooping is enabled at the global level.
- MLD snooping is not enabled and needs to be configured at the global level.

When you enable IGMP or MLD snooping over VPLS, traffic is forwarded on pseudowires that receive IGMP or MLD reports from remote Provider Edge (PE) devices. IGMP or MLD queries and reports are flooded to all the pseudowires.

For more information on:

- IGMP snooping, see *Configuring IGMP* in the *IP Multicast Routing Configuration Guide*.
- MLD snooping, see *Configuring MLD Snooping* in the *IP Multicast Routing Configuration Guide*.

How to Configure VPLS, VPLS BGP-Based Autodiscovery and Flow-Aware Transport

The following sections provide configuration information about VPLS, VPLS BGP-based autodiscovery and flow-aware transport.

Configuring Layer 2 PE Device Interfaces to CE Devices

You must configure Layer 2 PE device interfaces to CE devices. The following sections provide various configuration tasks that need to be completed before configuring VPLS.

Configuring 802.1Q Trunks on a PE Device for Tagged Traffic from a CE Device

To configure 802.1Q trunks on a PE device, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface TenGigabitEthernet1/0/24`	Defines the interface to be configured as a trunk, and enters interface configuration mode.
Step 4	**no ip address** *ip_address mask* [**secondary**] **Example:** `Device(config-if)# no ip address`	Disables IP processing and enters interface configuration mode.
Step 5	**switchport** **Example:** `Device(config-if)# switchport`	Modifies the switching characteristics of the Layer 2 switched interface.
Step 6	**switchport trunk encapsulation dot1q** **Example:** `Device(config-if)# switchport trunk encapsulation dot1q`	Sets the switch port encapsulation format to 802.1Q.
Step 7	**switchport trunk allow vlan** *vlan_ID* **Example:** `Device(config-if)# switchport trunk allow vlan 2129`	Sets the list of allowed VLANs.
Step 8	**switchport mode trunk** **Example:** `Device(config-if)# switchport mode trunk`	Sets the interface to a trunking VLAN Layer 2 interface.
Step 9	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Configuring 802.1Q Access Ports on a PE Device for Untagged Traffic from a CE Device

To configure 802.1Q access ports on a PE device, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface TenGigabitEthernet1/0/24`	Defines the interface to be configured as a trunk, and enters interface configuration mode.
Step 4	**no ip address** *ip_address mask* [**secondary**] **Example:** `Device(config-if)# no ip address`	Disables IP processing.
Step 5	**switchport** **Example:** `Device(config-if)# switchport`	Modifies the switching characteristics of the Layer 2 switched interface.
Step 6	**switchport mode access** **Example:** `Device(config-if)# switchport mode access`	Sets the interface type to nontrunking and nontagged single VLAN Layer 2 interface.
Step 7	**switchport access vlan** *vlan_ID* **Example:** `Device(config-if)# switchport access vlan 2129`	Sets the VLAN when the interface is in access mode.
Step 8	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Configuring Layer 2 VLAN Instances on a PE Device

Configuring the Layer 2 VLAN interface on the PE device, enables the Layer 2 VLAN instance on the PE device to the VLAN database, to set up the mapping between the VPLS and VLANs.

To configure Layer 2 VLAN instance on a PE device, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**vlan** *vlan-id* **Example:** `Device(config)# vlan 2129`	Configures a specific VLAN.
Step 4	**interface vlan** *vlan-id* **Example:** `Device(config-vlan)# interface vlan 2129`	Configures an interface on the VLAN.
Step 5	**end** **Example:** `Device(config-vlan)# end`	Returns to privileged EXEC mode.

Configuring VPLS

VPLS can be configured using either the Xconnect mode or protocol-CLI method.The following sections provide information about how to configure VPLS.

Configuring VPLS in Xconnect Mode

The following sections provide information on configuring VPLS in Xconnect mode.

Configuring MPLS on a PE Device

To configure MPLS on a PE device, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**mpls ip** **Example:** `Device(config)# mpls ip`	Configures MPLS hop-by-hop forwarding.
Step 4	**mpls label protocol ldp** **Example:** `Device(config)# mpls label protocol ldp`	Specifies the default Label Distribution Protocol (LDP) for a platform.
Step 5	**mpls ldp logging neighbor-changes** **Example:** `Device(config)# mpls ldp logging neighbor-changes`	(Optional) Determines logging neighbor changes.
Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

Configuring VFI on a PE Device

The VFI specifies the VPN ID of a VPLS domain, the addresses of other PE devices in this domain, and the type of tunnel signaling and encapsulation mechanism for each peer device.

To configure VFI and associated VCs on the PE device, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:**	Enables privileged EXEC mode. Enter your password if prompted.

	Command or Action	Purpose
	`Device> enable`	
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2 vfi** *vfi-name* **manual** **Example:** `Device(config)# l2 vfi 2129 manual`	Enables the Layer 2 VFI manual configuration mode.
Step 4	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 2129`	Configures a VPN ID for a VPLS domain. The emulated VCs bound to this Layer 2 virtual routing and forwarding (VRF) use this VPN ID for signaling. **Note** *vpn-id* is the same as *vlan-id*.
Step 5	**neighbor** *router-id* {**encapsulation mpls**} **Example:** `Device(config-vfi)# neighbor remote-router-id encapsulation mpls`	Specifies the remote peering router ID and the tunnel encapsulation type or the pseudowire (PW) property to be used to set up the emulated VC.
Step 6	**end** **Example:** `Device(config-vfi)# end`	Returns to privileged EXEC mode.

Associating the Attachment Circuit with the VFI on the PE Device

After defining the VFI, you must associate it to one or more attachment circuits.

To associate the attachment circuit with the VFI, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface vlan** *vlan-id* **Example:** `Device(config)# interface vlan 2129`	Creates or accesses a dynamic switched virtual interface (SVI). **Note** *vlan-id* is the same as *vpn-id*.
Step 4	**no ip address** **Example:** `Device(config-if)# no ip address`	Disables IP processing. (You can configure a Layer 3 interface for the VLAN if you need to configure an IP address.)
Step 5	**xconnect vfi** *vfi-name* **Example:** `Device(config-if)# xconnect vfi 2129`	Specifies the Layer 2 VFI that you are binding to the VLAN port.
Step 6	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Configuring VPLS in Protocol-CLI Mode

The following sections provide information on configuring VPLS in protocol-CLI mode.

Configuring VPLS in Protocol-CLI Mode

To configure VPLS in protocol-CLI mode, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	
Step 3	**l2vpn vfi context** *vfi-name* **Example:** `Device(config)# l2vpn vfi context vpls1`	Establishes an Layer 2 VPN VFI context and enters Layer 2 VFI configuration mode.
Step 4	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 10`	Configures a VPN ID for the VPLS domain.
Step 5	**member** *ip-address* **encapsulation mpls** **Example:** `Device(config-vfi)# member 2.2.2.2 encapsulation mpls`	Specifies the devices that form a point-to-point Layer 2 VPN VFI connection.
Step 6	**exit** **Example:** `Device(config-vfi)# exit`	Exits to privileged EXEC mode.

Step 7	Select one of the following:
	• **vlan configuration** *vlan-id*
	• **interface vlan** *vlan-id*
	Example:
	`Device(config)# vlan configuration 100`
	`OR`
	`Device(config)# interface vlan 100`
		Applies configuration to be applied on the VLAN or interface and enters VLAN or interface configuration mode.

Step 8	**member vfi** *vfi-name* **Example:** `Device(config-vlan-config)# member vfi vpls1`	Binds a VFI instance to a VLAN or an interface.
Step 9	**end** **Example:** `Device(config-vlan-config)# end`	Returns to privileged EXEC mode.

Configuring VPLS Flow-Aware Transport with Pseudowire Interface (in Protocol-CLI Mode)

To configure VPLS flow-aware transport with pseudowire interface, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface pseudowire** *number* **Example:** `Device(config)# interface pseudowire 1001`	Establishes a PW with the specified name, and enters pseudowire interface configuration mode.
Step 4	**encapsulation mpls** **Example:** `Device(config-if)# encapsulation mpls`	Specifies the tunneling encapsulation as MPLS.
Step 5	**neighbor** *peer-address vcid-value* **Example:** `Device(config-if)# neighbor 10.1.1.200 200`	Specifies the peer IP address and VC ID value of a Layer 2 VPN PW.
Step 6	**load-balance flow** **Example:** `Device(config-if)# load-balance flow`	Enables the load balancing with PW feature so that load balancing is done on a per-flow basis.
Step 7	**load-balance flow-label** **Example:** `Device(config-if)# load-balance flow-label both`	Enables the flow-aware transport of MPLS PW feature and specifies how flow labels are to be used.
Step 8	**exit** **Example:** `Device(config-if)# exit`	Exits to privileged EXEC mode.
Step 9	**l2vpn vfi context** *vfi-name* **Example:**	Establishes an Layer 2 VPN VFI context and enters Layer 2 VFI configuration mode.

	Command or Action	Purpose
	`Device(config)# l2vpn vfi context vpls1`	
Step 10	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 10`	Configures a VPN ID for the VPLS domain.
Step 11	**member pseudowire** *number* **Example:** `Device(config-vfi)# member pseudowire 1001`	Adds the pseudowire interface as a member of the VFI.
Step 12	**exit** **Example:** `Device(config-vfi)# exit`	Exits to privileged EXEC mode.

Step 13	Select one of the following:
	• **vlan configuration** *vlan-id*
	• **interface vlan** *vlan-id*
	Example:
	`Device(config)# vlan configuration 100`
	`OR`
	`Device(config)# interface vlan 100`
		Applies configuration to be applied on the VLAN or interface and enters VLAN or interface configuration mode.

Step 14	**member vfi** *vfi-name* **Example:** `Device(config-vlan-config)# member vfi vpls1`	Binds a VFI instance to a VLAN or an interface.
Step 15	**end** **Example:** `Device(config-vlan-config)# end`	Returns to privileged EXEC mode.

Configuring VPLS Flow-Aware Transport Using a Template (in Protocol-CLI Mode)

Configuring VPLS flow-aware transport using a template allows multiple PWs to share the same configuration.

To configure VPLS flow-aware transport using a template, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**template type pseudowire** [*template-name*] **Example:** `Device(config)# template type pseudowire mpls`	Specifies the name of a Layer 2 PW and enters pseudowire template configuration mode.
Step 4	**encapsulation mpls** **Example:** `Device(config-template)# encapsulation mpls`	Specifies the tunneling encapsulation as MPLS.
Step 5	**load-balance flow** **Example:** `Device(config-template)# load-balance flow`	Enables the load balancing with PW feature so that load balancing is done on a per-flow basis.
Step 6	**load-balance flow-label** **Example:** `Device(config-template)# load-balance flow-label both`	Enables the flow-aware transport of MPLS PW feature and specifies how flow labels are to be used.
Step 7	**exit** **Example:** `Device(config-template)# exit`	Exits to privileged EXEC mode.
Step 8	**l2vpn vfi context** *vfi-name* **Example:** `Device(config)# l2vpn vfi context vpls1`	Establishes an Layer 2 VPN VFI context and enters Layer 2 VFI configuration mode.

	Command or Action	Purpose
Step 9	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 10`	Configures a VPN ID for the VPLS domain.
Step 10	**member** *ip-address* **template** *template-name* **Example:** `Device(config-vfi)# member 102.102.102.102 template mpls`	Specifies the devices that form a point-to-point Layer 2 VPN VFI connection. • **ip-address**: IP address of the VFI neighbor. • **template** *template-name*: Specifies the template name mpls as the template method.
Step 11	**exit** **Example:** `Device(config-vfi)# exit`	Exits to privileged EXEC mode.

Step 12	Select one of the following:
	• **vlan configuration** *vlan-id*
	• **interface vlan** *vlan-id*
	Example:
	`Device(config)# vlan configuration 100`
	`OR`
	`Device(config)# interface vlan 100`
		Applies configuration to be applied on the VLAN or interface and enters VLAN or interface configuration mode.

Step 13	**member vfi** *vfi-name* **Example:** `Device(config-vlan-config)# member vfi vpls1`	Binds a VFI instance to a VLAN or an interface.
Step 14	**end** **Example:** `Device(config-vlan-config)# end`	Exits to privileged EXEC mode.

Configuring VPLS Flow-Aware Transport Using Pseudowire and a Template (in Protocol-CLI Mode)

To configure VPLS flow-aware transport using both PW and a template, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**template type pseudowire** [*template-name*] **Example:** `Device(config)# template type pseudowire mpls`	Specifies the name of a Layer 2 PW and enters pseudowire template configuration mode.
Step 4	**encapsulation mpls** **Example:** `Device(config-template)# encapsulation mpls`	Specifies the tunneling encapsulation as MPLS.
Step 5	**load-balance flow** **Example:** `Device(config-template)# load-balance flow`	Enables the load balancing with PW feature so that load balancing is done on a per-flow basis.
Step 6	**load-balance flow-label** **Example:** `Device(config-template)# load-balance flow-label both`	Enables the flow-aware transport of MPLS PW feature and specifies how flow labels are to be used.
Step 7	**exit** **Example:** `Device(config-template)# exit`	Exits to privileged EXEC mode.
Step 8	**interface pseudowire** *number* **Example:** `Device(config)# interface pseudowire 1001`	Establishes a PW with the specified name, and enters pseudowire interface configuration mode.

	Command or Action	Purpose
Step 9	**source template type pseudowire** [*template-name*] **Example:** `Device(config-if)# source template type pseudowire mpls`	Configures the source template of type pseudowire named mpls.
Step 10	**neighbor** *peer-address vcid-value* **Example:** `Device(config-if)# neighbor 10.1.1.200 200`	Specifies the peer IP address and VC ID value of a Layer 2 VPN PW.
Step 11	**exit** **Example:** `Device(config-if)# exit`	Exits to privileged EXEC mode.
Step 12	**l2vpn vfi context** *vfi-name* **Example:** `Device(config)# l2vpn vfi context vpls1`	Establishes an Layer 2 VPN VFI context and enters Layer 2 VFI configuration mode.
Step 13	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 10`	Configures a VPN ID for the VPLS domain.
Step 14	**member pseudowire** *number* **Example:** `Device(config-vfi)# member pseudowire 1001`	Adds the pseudowire interface as a member of the VFI.
Step 15	**exit** **Example:** `Device(config-vfi)# exit`	Exits to privileged EXEC mode.

Step 16	Select one of the following:
	• **vlan configuration** *vlan-id*
	• **interface vlan** *vlan-id*
	Example:
	`Device(config)# vlan configuration 100`
	OR
	`Device(config)# interface vlan 100`
		Applies configuration to be applied on the VLAN or interface and enters VLAN or interface configuration mode.

	Command or Action	Purpose
Step 17	**member vfi** *vfi-name* **Example:** `Device(config-vlan-config)# member vfi vpls1`	Binds a VFI instance to a VLAN or an interface.
Step 18	**end** **Example:** `Device(config-vlan-config)# end`	Exits to privileged EXEC mode.

Configuring VPLS BGP-based Autodiscovery

The following sections provide information about how to configure VPLS BGP-based Autodiscovery.

Enabling VPLS BGP-based Autodiscovery

To enabling VPLS BGP-based autodiscovery, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2 vfi** *vfi-name* **autodiscovery** **Example:** `Device(config)# l2 vfi 2128 autodiscovery`	Enables VPLS autodiscovery on a PE device and enters L2 VFI configuration mode.
Step 4	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 2128`	Configures a VPN ID for the VPLS domain.

	Command or Action	Purpose
Step 5	**end** **Example:** `Device(config-vfi)# end`	Returns to privileged EXEC mode.

Configuring BGP to Enable VPLS Autodiscovery

To configure BGP to enable VPLS autodiscovery, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 1000`	Enters router configuration mode for the specified routing process.
Step 4	**no bgp default ipv4-unicast** **Example:** `Device(config-router)# no bgp default ipv4-unicast`	Disables the IPv4 unicast address family for the BGP routing process. **Note** Routing information for the IPv4 unicast address family is advertised by default for each BGP routing session configured using the **neighbor remote-as router** command unless you configure the **no bgp default ipv4-unicast** command before configuring the **neighbor remote-as** command. Existing neighbor configurations are not affected.
Step 5	**bgp log-neighbor-changes** **Example:**	Enables logging of BGP neighbor resets.

	Command or Action	Purpose
	`Device(config-router)# bgp log-neighbor-changes`	
Step 6	**neighbor remote-as** { *ip-address* | *peer-group-name* } **remote-as** *autonomous-system-number* **Example:** `Device(config-router)# neighbor 44.254.44.44 remote-as 1000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local device. • If the *autonomous-system-number* argument matches the autonomous system number specified in the **router bgp** command, the neighbor is an internal neighbor. • If the *autonomous-system-number* argument does not match the autonomous system number specified in the **router bgp** command, the neighbor is an external neighbor.
Step 7	**neighbor** { *ip-address* | *peer-group-name* } **update-source** *interface-type interface-number* **Example:** `Device(config-router)# neighbor 44.254.44.44 update-source Loopback300`	(Optional) Configures a device to select a specific source or interface to receive routing table updates.
Step 8	Repeat Steps 6 and 7 to configure other BGP neighbors.	Exits interface configuration mode.
Step 9	**address-family l2vpn** [**vpls**] **Example:** `Device(config-router)# address-family l2vpn vpls`	Specifies the Layer 2 VPN address family and enters address family configuration mode. The optional **vpls** keyword specifies that the VPLS endpoint provisioning information is to be distributed to BGP peers.
Step 10	**neighbor** { *ip-address* | *peer-group-name* } **activate** **Example:** `Device(config-router-af)# neighbor 44.254.44.44 activate`	Enables the exchange of information with a BGP neighbor.
Step 11	**neighbor** { *ip-address* | *peer-group-name* } **send-community** { **both** | **standard** | **extended** } **Example:**	Specifies that a communities attribute should be sent to a BGP neighbor.

	Command or Action	Purpose
	`Device(config-router-af)# neighbor 44.254.44.44 send-community both`	
Step 12	Repeat Steps 10 and 11 to activate other BGP neighbors under an L2VPN address family.	
Step 13	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits address family configuration mode and returns to router configuration mode.
Step 14	**end** **Example:** `Device(config-router)# end`	Exits router configuration mode and returns to privileged EXEC mode.

Configuring VPLS BGP-based Autodiscovery in Protocol-CLI Mode

The following sections provide information on configuring VPLS BGP-based autodiscovery in protocol-CLI mode.

Configuring VPLS BGP based Autodiscovery in Protocol-CLI mode

To configure VPLS BGP based autodiscovery in protocol-CLI mode, perform this procedure

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2vpn vfi context** *vfi-name* **Example:** `Device(config)# l2vpn vfi context vpls1`	Establishes an Layer 2 VPN VFI context and enters Layer 2 VFI configuration mode.

	Command or Action	Purpose
Step 4	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 10`	Configures a VPN ID for the VPLS domain.
Step 5	**autodiscovery bgp signaling ldp** **Example:** `Device(config-vfi)# autodiscovery bgp signaling ldp`	Enables BGP signaling and LDP signaling.
Step 6	**exit** **Example:** `Device(config-vfi-autodiscovery)# exit`	Exits to privileged EXEC mode.
Step 7	**exit** **Example:** `Device(config-vfi)# exit`	Exits to privileged EXEC mode.

Step 8	Select one of the following:
	• **vlan configuration** *vlan-id*
	• **interface vlan** *vlan-id*
	Example:
	`Device(config)# vlan configuration 100`
	OR
	`Device(config)# interface vlan 100`
		Applies configuration to be applied on the VLAN or interface and enters VLAN or interface configuration mode.

Step 9	**member vfi** *vfi-name* **Example:** `Device(config-vlan-config)# member vfi vpls1`	Binds a VFI instance to a VLAN or an interface.
Step 10	**end** **Example:** `Device(config-vlan-config)# end`	Exits to privileged EXEC mode.

Configuring VPLS BGP based Autodiscovery Flow-Aware Transport using Template (in Protocol-CLI Mode)

To configure VPLS BGP based autodiscovery flow-aware transport using template, perform this procedure

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**template type pseudowire** [*template-name*] **Example:** `Device(config)# template type pseudowire mpls`	Specifies the name of a Layer 2 PW and enters pseudowire template configuration mode.
Step 4	**encapsulation mpls** **Example:** `Device(config-template)# encapsulation mpls`	Specifies the tunneling encapsulation as MPLS.
Step 5	**load-balance flow** **Example:** `Device(config-template)# load-balance flow`	Enables the Any Transport over MPLS (AToM) load balancing with PW feature so that load balancing is done on a per-flow basis.
Step 6	**load-balance flow-label** **Example:** `Device(config-template)# load-balance flow-label both`	Enables the flow-aware transport of MPLS PW feature and specifies how flow labels are to be used.
Step 7	**exit** **Example:** `Device(config-template)# exit`	Exits to privileged EXEC mode.
Step 8	**l2vpn vfi context** *vfi-name* **Example:** `Device(config)# l2vpn vfi context vpls1`	Establishes an Layer 2 VPN VFI context and enters Layer 2 VFI configuration mode.

	Command or Action	Purpose
Step 9	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 10`	Configures a VPN ID for the VPLS domain.
Step 10	**autodiscovery bgp signaling ldp template** *name* **Example:** `Device(config-vfi)# autodiscovery bgp signaling ldp template mpls`	Enables BGP signaling and LDP signaling.
Step 11	**exit** **Example:** `Device(config-vfi)# exit`	Exits to privileged EXEC mode.

Step 12	Select one of the following:
	• **vlan configuration** *vlan-id*
	• **interface vlan** *vlan-id*
	Example:
	`Device(config)# vlan configuration 100`
	`OR`
	`Device(config)# interface vlan 100`
		Applies configuration to be applied on the VLAN or interface and enters VLAN or interface configuration mode.

Step 13	**member vfi** *vfi-name* **Example:** `Device(config-vlan-config)# member vfi vpls1`	Binds a VFI instance to a VLAN or an interface.
Step 14	**end** **Example:** `Device(config-vlan-config)# end`	Exits to privileged EXEC mode.

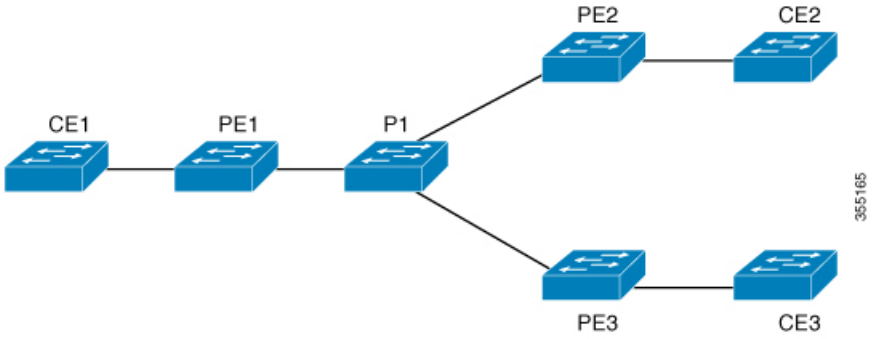
Configuration Examples for VPLS and VPLS BGP-Based Autodiscovery

This section provides the configuration examples for VPLS and VPLS BGP-Based Autodiscovery.

Example: Configuring VPLS in Xconnect Mode

The following example shows how to configure VPLS on a PE1 and PE2 devices:

Figure 3: VPLS Topology



PE1 Configuration

```
Device> enable
Device# configure terminal
Device(config)# pseudowire-class vpls2129
Device(config-if)# encapsulation mpls
Device(config-if)# exit
Device(config)# l2 vfi 2129 manual
Device(config-vfi)# vpn id 2129
Device(config-vfi)# neighbor 44.254.44.44 pw-class vpls2129
Device(config-vfi)# neighbor 188.98.89.98 pw-class vpls2129
Device(config-vfi)# exit
Device(config)# interface TenGigabitEthernet1/0/24
Device(config-if)# switchport trunk allowed vlan 2129
Device(config-if)# switchport mode trunk
Device(config-if)# exit
Device(config)# interface vlan 2129
Device(config-vlan-config)# no ip address
Device(config-vlan-config)# xconnect vfi 2129
```

Examples: Verifying VPLS Configured in Xconnect Mode

The following example is a sample output of the **show mpls l2transport vc detail** command. This command provides information about the virtual circuits.

```
Device# show mpls l2transport vc detail
 Local interface: VFI 2129 vfi up
  Interworking type is Ethernet
  Destination address: 44.254.44.44, VC ID: 2129, VC status: up
    Output interface: Gi1/0/9, imposed label stack {18 17}
    Preferred path: not configured
    Default path: active
    Next hop: 177.77.177.2
  Create time: 19:09:33, last status change time: 09:24:14
    Last label FSM state change time: 09:24:14
```

```
  Signaling protocol: LDP, peer 44.254.44.44:0 up
    Targeted Hello: 1.1.1.72(LDP Id) -> 44.254.44.44, LDP is UP
    Graceful restart: configured and enabled
    Non stop routing: not configured and not enabled
    Status TLV support (local/remote)   : enabled/supported
      LDP route watch                   : enabled
      Label/status state machine        : established, LruRru
      Last local dataplane   status rcvd: No fault
Last BFD dataplane     status rcvd: Not sent
      Last BFD peer monitor  status rcvd: No fault
      Last local AC  circuit status rcvd: No fault
      Last local AC  circuit status sent: No fault
      Last local PW i/f circ status rcvd: No fault
      Last local LDP TLV     status sent: No fault
      Last remote LDP TLV    status rcvd: No fault
      Last remote LDP ADJ    status rcvd: No fault
MPLS VC labels: local 512, remote 17
    Group ID: local n/a, remote 0
    MTU: local 1500, remote 1500
    Remote interface description:
  Sequencing: receive disabled, send disabled
  Control Word: Off
  SSO Descriptor: 44.254.44.44/2129, local label: 512
  Dataplane:
    SSM segment/switch IDs: 20498/20492 (used), PWID: 2
  VC statistics:
    transit packet totals: receive 0, send 0
    transit byte totals:   receive 0, send 0
    transit packet drops:  receive 0, seq error 0, send 0
```

The following example is a sample output of the **show l2vpn atom vc** command. The command shows that AToM over MPLS is configured on a VC.

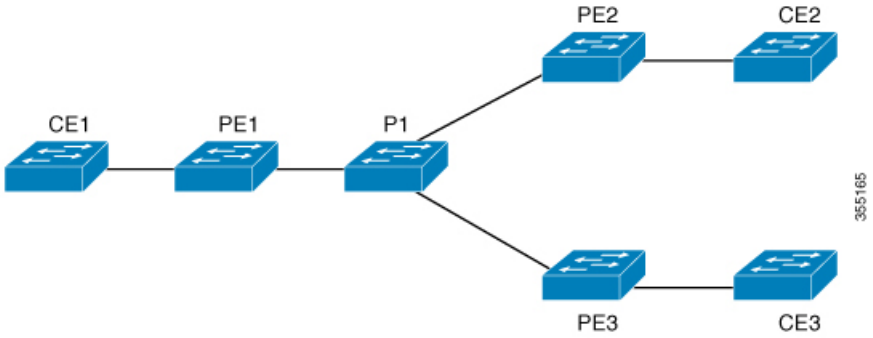
```
Device# show l2vpn atom vc detail

pseudowire100005 is up, VC status is up PW type: Ethernet
  Create time: 19:25:56, last status change time: 09:40:37
    Last label FSM state change time: 09:40:37
  Destination address: 44.254.44.44 VC ID: 2129
    Output interface: Gi1/0/9, imposed label stack {18 17}
    Preferred path: not configured
    Default path: active
    Next hop: 177.77.177.2
  Member of vfi service 2129
    Bridge-Domain id: 2129
    Service id: 0x32000003
  Signaling protocol: LDP, peer 44.254.44.44:0 up
    Targeted Hello: 1.1.1.72(LDP Id) -> 44.254.44.44, LDP is UP
    Graceful restart: configured and enabled
    Non stop routing: not configured and not enabled
     PWid FEC (128), VC ID: 2129
    Status TLV support (local/remote)         : enabled/supported
      LDP route watch                         : enabled
      Label/status state machine              : established, LruRru
      Local dataplane status received         : No fault
      BFD dataplane status received           : Not sent
      BFD peer monitor status received        : No fault
      Status received from access circuit     : No fault
      Status sent to access circuit           : No fault
      Status received from pseudowire i/f     : No fault
Status sent to network peer             : No fault
```

```
    Status received from network peer       : No fault
    Adjacency status of remote peer         : No fault
Sequencing: receive disabled, send disabled
Bindings
  Parameter    Local                          Remote
  ------------ ------------------------------ ------------------------------
  Label        512                            17
  Group ID     n/a                            0
  Interface
  MTU          1500                           1500
  Control word off                            off
  PW type      Ethernet                       Ethernet
  VCCV CV type 0x02                           0x02
                 LSPV [2]                       LSPV [2]
  VCCV CC type 0x06                           0x06
                 RA [2], TTL [3]                RA [2], TTL [3]
  Status TLV   enabled                        supported
SSO Descriptor: 44.254.44.44/2129, local label: 512
Dataplane:
  SSM segment/switch IDs: 20498/20492 (used), PWID: 2
Rx Counters
  0 input transit packets, 0 bytes
  0 drops, 0 seq err
Tx Counters
  0 output transit packets, 0 bytes
  0 drops
```

Example: Configuring VPLS Flow-Aware Transport Using a Template (in Protocol-CLI Mode)

The following example shows how to configure VPLS on a PE1 and PE2 devices:

Figure 4: VPLS Topology



PE1 Configuration

```
Device> enable
Device# configure terminal
Device(config)# template type pseudowire mpls
Device(config-template)# encapsulation mpls
Device(config-template)# load-balance flow ip dst-ip
Device(config-template)# load-balance flow-label both
Device(config-template)# exit
Device(config)# interface Loopback0
Device(config-if)# ip address 1.1.1.30 255.255.255.255
Device(config-if)# ip ospf 1 area 0
Device(config-if)# exit
Device(config)# interface TwentyFiveGigE1/0/9
Device(config-if)# no switchport
Device(config-if)# ip address 80.0.0.30 255.255.255.0
Device(config-if)# ip ospf 1 area 0
Device(config-if)# mpls ip
Device(config-if)# exit
Device(config)# l2vpn vfi context foo
Device(config-vfi)# vpn id 2129
Device(config-vfi)# member 1.1.1.20 template mpls
Device(config-vfi)# exit
Device(config)# interface TwentyFiveGigE1/0/2
Device(config-if)# switchport mode access
Device(config-if)# switchport access vlan 100
Device(config-if)# exit
Device(config)# interface vlan 100
Device(config-vlan-config)# member vfi foo
Device(config-vlan-config)# end
```

Example: Configuring VPLS BGP-Auto Discovery

The following example shows how to configure VPLS on a PE device:

```
Device> enable
Device# configure terminal
Device(config)# router bgp 1000
Device(config-router)# bgp log-neighbor-changes
Device(config-router)# bgp graceful-restart
Device(config-router)# neighbor 44.254.44.44 remote-as 1000
Device(config-router)# neighbor 44.254.44.44 update-source Loopback300
Device(config-router)# address-family l2vpn vpls
Device(config-router-af)# neighbor 44.254.44.44 activate
Device(config-router-af)# neighbor 44.254.44.44 send-community both
Device(config-router-af)# exit-address-family
Device(config-router-af)# end
Device(config)# l2 vfi 2128 autodiscovery
Device(config-vfi)# vpn id 2128
Device(config-vfi)# exit
Device(config)# interface vlan 2128
Device(config-vlan-config)# no ip address
Device(config-vlan-config)# xconnect vfi 2128
!
```

Example: Verifying VPLS BGP-Auto Discovery

The following example is a sample output of the **show platform software fed sw 1 matm macTable vlan 2000** command.

```
Device# show platform software fed sw 1 matm macTable vlan 2000

VLAN  MAC              Type      Seq#   macHandle          siHandle           diHandle
       *a_time *e_time   ports
2000  2852.6134.05c8   0X8002    0      0xffbba312c8       0xffbb9ef938       0x5154
       0        0        Vlan2000
2000  0000.0078.9012   0X1       32627  0xffbb665ec8       0xffbb60b198       0xffbb653f98
       300      278448   Port-channel11
2000  2852.6134.0000   0X1       32651  0xffba15e1a8       0xff454c2328       0xffbb653f98
       300      63       Port-channel11
2000  0000.0012.3456   0X2000001 32655  0xffba15c508       0xff44f9ec98       0x0
       300      1        2000:33.33.33.33
Total Mac number of addresses:: 4
*a_time=aging_time(secs)  *e_time=total_elapsed_time(secs)
Type:
MAT_DYNAMIC_ADDR     0x1       MAT_STATIC_ADDR       0x2
MAT_CPU_ADDR         0x4       MAT_DISCARD_ADDR      0x8
MAT_ALL_VLANS        0x10      MAT_NO_FORWARD        0x20
MAT_IPMULT_ADDR      0x40      MAT_RESYNC            0x80
MAT_DO_NOT_AGE       0x100     MAT_SECURE_ADDR       0x200
MAT_NO_PORT          0x400     MAT_DROP_ADDR         0x800
MAT_DUP_ADDR         0x1000    MAT_NULL_DESTINATION  0x2000
MAT_DOT1X_ADDR       0x4000    MAT_ROUTER_ADDR       0x8000
MAT_WIRELESS_ADDR    0x10000   MAT_SECURE_CFG_ADDR   0x20000
MAT_OPQ_DATA_PRESENT 0x40000   MAT_WIRED_TUNNEL_ADDR 0x80000
MAT_DLR_ADDR         0x100000  MAT_MRP_ADDR          0x200000
MAT_MSRP_ADDR        0x400000  MAT_LISP_LOCAL_ADDR   0x800000
MAT_LISP_REMOTE_ADDR 0x1000000 MAT_VPLS_ADDR         0x2000000
```

The following example is a sample output of the **show bgp l2vpn vpls all** command.

```
Device# show bgp l2vpn vpls all

BGP table version is 6, local router ID is 222.5.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
  r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
  x best-external, a additional-path, c RIB-compressed,
  t secondary path,
 Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 1000:2128
*>   1000:2128:1.1.1.72/96
                 0.0.0.0                              32768 ?
*>i  1000:2128:44.254.44.44/96
                 44.254.44.44             0    100      0 ?
```

Feature History for VPLS and VPLS BGP-Based Autodiscovery

This table provides release and related information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Everest 16.5.1a	Configuring VPLS and VPLS BGP-based Autodiscovery	VPLS enables enterprises to link together their Ethernet-based LANs from multiple sites via the infrastructure provided by their service provider. VPLS Autodiscovery enables each PE device to discover other PE devices that are part of the same VPLS domain.
Cisco IOS XE Amsterdam 17.1.1	VPLS Layer 2 Snooping : IGMP (IPv4)	IGMP snooping is supported on a VPLS configured network.
Cisco IOS XE Bengaluru 17.6.1	MLD Snooping over VPLS	Support was introduced for MLD Snooping over VPLS. It allows traffic to be forwarded on pseudowires that receive IGMP/MLD reports from remote Provider Edge (PE) devices.

Use the Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to https://cfnng.cisco.com/

http://www.cisco.com/go/cfn.